[0074] Obligations

[0075] An obligation, or event pattern/response relation, defines a set of conditions and methods under which policy state data is dynamically obligated to change. An event pattern/response relation is a pair (ep, r) (usually denoted ep⇒r), where ep is an event pattern and r is a sequence of primitive administrative operations, called a response. The event pattern specifies conditions related to a process' successful execution of an operation on an object, using parameters like the user of the process, the operation executed, and the container(s) in which the object is included. The set of obligations is denoted as EP-R in FIG. **2**.

[0076] A successful completion of an operation on an object may trigger an event. The context of the event comprises the process identifier and its user identity, the operation, the object on which the operation was performed, the object's containers, etc. The system **20** starts processing the event by determining the event patterns in the entire set of obligations that are matched by the event. The match is performed by checking whether the event context satisfies the conditions specified in the event pattern. For all successful matches, the system **20** executes the response associated with the matched pattern. Note that the possible formal parameters of the administrative operations comprised in the response are replaced by the appropriate values extracted from the event context. Responses are obligations performed by the system **20**, and as such, their execution is not predicated on permissions.

[0077] The system **20** also includes a fixed set of functions that include function for user authentication, session management, presentation of accessible objects, reference mediation, and event-response processing.

[0078] Authentication

[0079] A user interaction with the system **20** begins with the user's authentication. Although authentication is included among the functions, the system's **20** specification does not dictate the method (e.g., password, tokens) by which authentication is performed. Upon authentication, a session is created where all processes included in the session are associated with the authenticated user.

[0080] A process may issue an access request on behalf of its user, or independent of its user. A process access request may be denoted by $\langle ops, o\rangle_p$, where $p \in P$, $ops \subseteq OP$ and $o \in O$.

[0081] Personal Object System (POS)

[0082] Following user authentication, the user may be presented with a Personal Object System (POS). The POS is a graphical presentation of the set of objects that are currently accessible to the user. The graphical presentation organizes accessible objects into the set of containers (object attributes) to which the objects belong and which are also accessible by the user. Remembering that objects are also object containers, an object container is accessible to a user if that user is authorized to perform the same operation op on the objects contained in that container within each policy class that container belongs to. As illustrated in FIG. **6**, an object container (or attribute) represented as $oa \in OA$ is accessible to a user $u \in U$ if $\exists op \in OP$, such that $\forall pc \in PC$ such that $oa \rightarrow^{+} pc$, $\exists ua \in UA$, $\exists ops \in 2^{OP}$, $\exists oa' \in OA$, such that $op \in ops$, $u \rightarrow^{+} ua \rightarrow ops \rightarrow oa'$, $ua \rightarrow^{+} pc$, and $oa \rightarrow^{*} oa' \rightarrow^{+} pc$. FIG. **7** illustrates a representation of alice's POS with respect to FIG. **3**.

[0083] The personal object system of a user u, denoted by POS_u, is a directed graph (V, E) where the node set V is defined as follows:

[0084] 1. An object attribute oa is in V if and only if oa is in a policy class and oa is accessible to user u.

[0085] 2. A policy class pc is in V if and only if pc contains an object attribute accessible to u.

[0086] 3. No other node is in V.

[0087] and the arc set E is defined as follows:

[0088] 4. If policy class pc and object attribute oa are nodes in V, there is an arc from oa to pc if and only if oa is in pc (in the original graph) and there is no other object attribute in V on a path from oa to pc (in the original graph).

[0089] 5. If object attributes oa_1 and oa_2 are nodes in V, there is an arc from oa_1 to oa_2 if and only if there is a path from oa_1 to oa_2 (in the original graph) and there is no other object attribute in V on a path from oa_1 to oa_2 (in the original graph).

[0090] 6. No other arc is in E.

[0091] Reference Mediation

[0092] Either through the use of the POS or some other means of referencing objects, a user may issue a request to perform a set of operations on a set of objects, through a process. A process may also issue an access request without the intervention of a user. The server module **28** either grants or denies a process access request. A process access request to perform an operation op on an object o, with p being the process identifier, is granted if and only if there exists a permission (u, op, o) where u=process_user(p), and (op, o) is not denied (through prohibitions) for either p or u. We refer to this function of granting or denying a process access request as reference mediation.

[0093] Given process $p \in P$, user u=process_user(p), operation $op \in OP$, and object $o \in O$, reference_mediation($\langle op, o\rangle_p$)=grant $\overset{def}{\Leftrightarrow}$

[0094] 1. (u, op, o) is a permission ∧

[0095] 2. $\forall \langle u, ops, os\rangle \in UDENY, \neg(op \in ops \wedge o \in os) \wedge$

[0096] 3. $\forall \langle p, ops, os\rangle \in PDENY, \neg(op \in ops \wedge o \in os)$.

[0097] With respect to the definition of the capabilities and permissions in the system **20**, the reference mediation function grants a process p the permission to execute a request $\langle op, o\rangle_p$ if and only if, in each policy class that contains the object o, the pair (op, o) is a capability of an attribute of user u=process_user(p), and in addition, this capability is not prohibited by a deny relation.

[0098] Transferring Data Between Processes

[0099] Underlying resource management systems, on which access control depends, provide facilities for interprocess communication, and as such offer opportunities to "leak" data in a manner that may undermine the policy. For example, operating systems provide mechanisms for facilitating communications and data sharing between applications. These mechanisms include but are not limited to clipboards, pipes, sockets, remote procedure calls, and messages. They all conform to a common abstraction: one process produces/creates data and inserts it into the mechanism's physical medium; the other process consumes/reads the data from the physical medium. A synchronization mechanism must also exist.